2005

Simulation study of routing protocols in wireless sensor networks

Vatsalya Kunchakarra
Louisiana State University and Agricultural and Mechanical College, vkunch1@lsu.edu

SIMULATION STUDY OF ROUTING PROTOCOLS IN WIRELESS SENSOR NETWORKS

A Thesis
Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Master of Science in Systems Science

in

The Department of Computer Science

By
Vatsalya Kunchakarra
B.E. Electronics and Communications Engineering,
Osmania University,2003
December, 2005

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to Dr. Arjan Durresi, my advisor, for his invaluable guidance and encouragement extended throughout the study. His tenacious supervision, helpful suggestion, patience and time deserve a special mention

I would like to express my appreciation to my committee members Dr. S.S.Iyengar, Dr.Bijaya B. Karki and Dr.Seung-Jong Park for their support and suggestions.

I would also like to thank my colleagues Vamsi Paruchuri, Cariappa Mallanda, Varma Indukuri and Ankur Suri and all other friends for their support and suggestions during the progress of this thesis.

I thank my parents Mr. Nagesh K and Mrs. Manjula K and my brother for their continued support and encouragement in the course of doing my Master's degree.

Last, but not the least, I would like to gratefully acknowledge Department of Computer Science, Louisiana State University for providing the resources and needs during the thesis.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS…….. …. ii

LIST OF TABLES……...v

LIST OF FIGURES……..vii

ABSTRACT…….viii

CHAPTER 1: INTRODUCTION……1
1.1 Overview of Wireless Sensor Networks **……**1
1.2 Thesis Outline……..3

CHAPTER 2: OVERVIEW OF THE EXISTING NETWORK SIMULATORS……...4

CHAPTER 3: SENSORSIMULATOR ARCHITECTURE……..10
3.1 SensorSimulator Framework……10
3.2 Design Approach of the SensorSimulator Framework……..13
3.2.1 Target Node……...14
3.2.2 Sensor Node……...15
3.2.3 Hardware Model……16
3.3 New Implementation……18

CHAPTER 4: COMPARATIVE STUDY OF DIRECTED DIFFUSION……20

CHAPTER 5: IMPLEMENTATION OF BPS……..25
5.1 Study of Routing Protocols in Wireless Sensor Networks……25
5.2 BPS…….....25
5.3 BPS Algorithm……...27
5.4 Effect of Threshold Th……...31
5.5 BPS Efficiency……...31

CHAPTER 6: IMPLEMENTATION OF ECP……..34
6.1 Problem Statement……35
6.1.1 Problem SET-K-CDS……36
6.2 Protocol Description……..37
6.3 Performance Evaluation……39

CHAPTER 7: IMPLEMENTATION OF RAW……42
7.1 The Protocol……...43
7.2 Description……43
7.2.1 Sleep State……..43
7.2.2 Awake/Setup……..44
7.2.3 Ready to Receive State……..44
7.2.4 Receive Transmit……...45

7.3 Performance of RAW……...45

CHAPTER 8: CONCLUSION AND FUTURE WORK……...49

REFERENCES…… 50

VITA……..52

LIST OF TABLES

Table 5.1 Effect of *Th* on the performance of BPS…….32

LIST OF FIGURES

Figure 2.1 Simple and Compound Modules in OMNeT++……9

Figure 3.1 Sensor Node Representation in a Network……...10

Figure 4.1 Comparison of Delivery Ratio……..23

Figure 4.2 Execution Time Comparison – Directed Diffusion Vs MAC 802.11……..24

Figure 4.3 Memory Usage – 10 queries…….24

Figure 5.1 Number of transmissions to cover an entire region for different areas……33

Figure 5.2 Number of transmissions for varying node densities and different areas……33

Figure 6.1 Number of Backbone Nodes for different network sizes and densities……...40

Figure 6.2 End-to-End Packet Delay…….41

Figure 7.1 State Transition Diagram……..44

Figure 7.2 Effect of Awake Time of Nodes with Traffic……..46

Figure 7.3 Effect on Latency with Traffic…….46

Figure 7.4 Effect of Schedule Period on the performance of protocol……..47

Figure 7.5 Performance of the protocol for various densities……47

ABSTRACT

Wireless sensor networks, a distributed network of sensor nodes perform critical tasks in many application areas such as target tracking in military applications, detection of catastrophic events, environment monitoring, health applications etc. The routing protocols developed for these distributed sensor networks need to be energy efficient and scalable. To create a better understanding of the performance of various routing protocols proposed it is very important to perform a detailed analysis of them. Network simulators enable us to study the performance and behavior of these protocols on various network topologies.

Many Sensor Network frameworks were developed to explore both the networking issues and the distributed computing aspects of wireless sensor networks. The current work of simulation study of routing protocols is done on SensorSimulator, a discrete event simulation framework developed at Sensor Networks Research Laboratory, LSU and on a popular event driven network simulator ns2 developed at UC Berkeley.

SensorSimulator is a discrete event simulation framework for sensor networks built over OMNeT++ (Objective Modular Network Test-bed in C++). This framework allows the user to debug and test software for distributed sensor networks. SensorSimulator allows developers and researchers in the area of Sensor Networks to investigate topological, phenomenological, networking, robustness and scaling issues, to explore arbitrary algorithms for distributed sensors, and to defeat those algorithms through simulated failure. The framework has modules for all the layers of a Sensor Network Protocol stack. This thesis is focused on the simulation and performance evaluation of various routing protocols on SensorSimulator and ns2.

The performance of the simulator is validated with a comparative study of Directed Diffusion Routing Protocol on both ns2 and SensorSimulator. Then the simulations are done to evaluate the performance of Optimized Broadcast Protocols for Sensor Networks, Efficient Coordination Protocol for Wireless Sensor Networks on SensorSimulator. Also a performance study of Random Asynchronous Wakeup protocol for Sensor Networks is done on ns2.

CHAPTER 1: INTRODUCTION

1.1 Overview of Wireless Sensor Networks

Wireless Sensor Networks (WSN) [1] consists of numerous tiny sensors deployed at high density in regions requiring surveillance and monitoring. The sensors are deployed at a cost much lower than the traditional wired sensor system. A large number of sensors deployed will enable for accurate measurements. A Sensor Node consists of one or more sensing elements (motion, temperature, pressure, etc.), a battery, and low power radio trans-receiver, microprocessor and limited memory, mobilizer (optional), a position finding system. An important aspect of such networks is that the nodes are unattended, have limited energy and the network topology is unknown. Many design challenges that arise in sensor networks are due to the limited resources they have and their deployment in hostile environments.

Sensor nodes are deployed in environments where it is impractical or infeasible for humans to interact or monitor them. These unattended nodes may have effect on the efficiency of many military and civil applications such as target field imaging, distributed computing, intrusion detection, security and tactical surveillance, inventory control, disaster management and detecting ambient conditions. Some applications require sensors to be small in size and have short transmission ranges to reduce the chances of detection. These size constraints cause further constraints on CPU speed, amount of memory, RF bandwidth and battery lifetime. Hence, efficient communication techniques are essential for increasing the lifetime and quality of data collection and decreasing the communication latency of such wireless devices.

Unlike the mobile ad hoc networks, sensor nodes are most likely to be stationary for the entire period of their lifetime. Even though the sensor nodes are fixed, the topology of the network can change. During periods of low activity, nodes may go to inactive sleep state, to conserve energy. When some nodes run out of battery power and die, new nodes may be added to the network. Although all nodes are initially equipped with equal energy, some nodes may experience higher activity as result of region they are located in. Communication pattern is intermittent and sensor applications are data-centric in nature. An important property of sensor networks is the need of the sensors to reliably disseminate the data to the sink or the base station within a time interval that allows the user or controller application to respond to the information in a timely manner, as out of date information is of no use and may lead to disastrous results.

Another important attribute is the scalability to the change in network size, node density and topology. Sensor networks are very dense as compared to mobile ad hoc and wired networks. This arises from the fact that the sensing range is lesser than the communication range and hence more nodes are needed to achieve sufficient sensing coverage. Sensor nodes are required to be resistant to failures and attacks.

Information routing is a very challenging task in Distributed Sensor Networks due to the inherent characteristics that distinguish these networks from other wireless or ad-hoc networks. The sensor nodes deployed in an adhoc manner need to be self-organizing as this kind of deployment requires system to form connections and cope with the resultant nodal distribution. Another important design issue in sensor networks is that sensor networks are application specific. Hence the application scenario demands the protocol design in a sensor network. Also, the data collected by sensor nodes is often redundant and needs to be exploited by routing protocols to improve energy and

bandwidth utilization. The proposed routing protocols for sensor networks should consider all the above issues for it to be very efficient. The algorithms developed need to be very energy efficient, scalable and increase the life of the network in the process.

The multitudes of design challenges imposed on Sensor Networks tend to be quite complex and usually defy the analytical methods that are quite effective for traditional networks. At current stage of technology very few Sensor Networks have come into existence. Although there are many unsolved research problems in this domain, actual deployment and study is infeasible. The only practical alternate to study Sensor Networks is through simulation, which can provide better insight to behavior and performance of various algorithms and protocols.

1.2 Thesis Outline

The current work is focused to study the performance and behavior of these routing protocols on various network topologies. The report begins with an introduction to Wireless Sensor Networks and the importance of routing protocols in various Sensor Network Applications. Chapter 2 gives an overview on the existing simulators and a brief description of OMNeT++ Framework. Chapter 3 gives an overview on the design of SensorSimulator Architecture and how various protocols can be added at different layers without much dependency. Chapter 4 gives the comparative study of SensorSimulator with ns2, with Directed Diffusion protocol at the network layer and IEEE 802.11with DCF at the MAC layer. Chapter 5 gives an overview on Optimized Flooding Protocol and its implementation in the Simulator. Chapter 6 extends the above broadcasting protocol to an Efficient Coordination Protocol for Wireless Sensor Networks. Chapter 7 gives the simulation analysis for Random Asynchronous Wakeup Protocol designed for Wireless Sensor Networks. All the chapters also have details on the experimental results.

CHAPTER 2: OVERVIEW OF THE EXISTING NETWORK SIMULATORS

Network simulators are very important for analyzing various protocols designed for a network (wired or wireless) and its necessity is very well known in the field of research. Especially, the research challenges in wireless sensor networks brought many open issues to network designers. The techniques used for analyzing the performance of any wireless networks are physical measurement, analytical methods and computer simulation. The constraints imposed on sensor networks, such as energy limitation, fault tolerance make the algorithms for sensor networks to be quite complex and usually defy analytical methods that have been fairly effective for traditional networks. And physical measurement is not possible because of the unsolved research problems in the field of sensor networks. Hence computer simulations appear to be the only feasible approach than anything else [2].

ns2, a widely used network simulator in the research community has the extended features to simulate Sensor Networks. It uses object-oriented design for the implementation of various modules of a sensor network [3]. There are modules for energy model, wireless channel, sensor channel which models dynamic inter-action between the physical environment and the sensor nodes. It also has implementations of few protocols that are under development for sensor networks. These include S-MAC, a Sensor MAC protocol at the MAC layer in a Sensor Node protocol stack, Directed Diffusion routing protocol with Geographic Routing. It also has a framework developed for Sensor Networks known as SensorSim which has the detailed implementation of a Sensor Node with a hardware model defining the hardware components of a sensor node and a software model defining the protocol stack of the node. But the project was stopped

in between and the developed software is not available anymore. The object oriented design of ns2 introduces unnecessary interdependence between modules and makes the addition of new protocols very difficult as it can be mastered only by experts in ns2 [2]. This extension might be easy for traditional networks but not for sensor networks where the protocols are not very dominant and it is very unlikely that a single algorithm will be optimal under various circumstances. Also various simulation studies show that the memory utilization of ns2 is very high and increases for very large simulations. Since the application areas in sensor networks require many number of sensor nodes in a sensor field, the simulations in ns2 take lot of memory. Also another disadvantage posed by ns2 comes from its open source nature. The documentation is often limited and out of date with the current release of the simulator. The problems can be solved with the help of dynamic news groups and going through the source code. Also, the consistency of code is lacking as it is developed by many users. There are no tools describe simulation scenarios and analyze or visualize simulation trace files. The tools for ns2 are written with scripting languages. The lack of generalized analysis tools may lead to different people measuring different values for the same metric names [12].

OPNET modeler is another popular commercial platform for network modeling and simulation which allows the design and study of communication networks, devices, protocols, and applications with unmatched flexibility and scalability. This is used by many prestigious technology organizations to accelerate the research and development process. OPNET Modeler is based on a series of hierarchical editors that directly parallel the structure of real networks, equipment, and protocols. The wireless model uses a 13-stage pipeline to determine connectivity and propagation among nodes. Modeler's object-

oriented modeling and hierarchical editors mirror the structure of actual networks and network components [4]. The difficulty with OPNET Modeler is to build the state machine for each level of the protocol stack. It is difficult to abstract such a state machine starting from a pseudo-coded algorithm. But state machines are the most practical input for discrete simulators. Hence, it is possible to reuse a lot of existing components (MAC layer, transceivers, links, etc.) improving the deployment process. But on the other hand, any new feature must be described as a finite state machine which can be difficult to debug, extend and validate [12]. Also it is commercial and is not available for public which becomes the biggest disadvantage for working on it.

J-Sim is an open-source, component based network simulation environment developed entirely in Java by Ohio State University (initially and later by University of Illinois). This along with the autonomous component architecture makes it a truly platform-neutral, extensible, and reusable environment. The Sensor Network Framework developed in J-Sim provides an object-oriented definition for target, sensor and sink nodes; sensor and wireless communication channels; and physical media such as seismic channels, mobility model and power model [5]. The simulation analysis described in [12] show that the execution speed of J-Sim is less compared to many other simulators and this happens because of its implementation in JAVA. But the memory consumption of H-Sim is less compared to others and this advantage comes from its garbage collectors.

GloMoSim developed initially at UCLA Computing Laboratory, is a scalable simulation environment for wireless and wired networks systems developed [2]. It is designed using the parallel discrete-event simulation capability provided by a C-based parallel simulation language, Parsec [12]. It currently supports protocols for purely wireless networks and is built using a layered approach. Standard APIs are used between

the different layers and allow the rapid integration of models developed at different layers by users. The difficulty with GloMoSim was to describe a simple application that bypasses most OSI layers. The bypass of the protocol stack is not obvious to achieve as most applications usually lie on top of it. The architecture is also not very flexible compared to other simulators.

Though many simulators were developed to emulate a Sensor Network, each has its own design complexities to test and verify new protocols.

The current study of routing protocols is done on SensorSimulator developed by the Sensor Networks Research Laboratory, LSU. SensorSimulator is developed to debug and test software for distributed sensor networks independent of hardware constraints. The extensibility of SensorSimulator allowed to investigate the topological, phenomenological, networking, robustness and scaling issues, of the proposed routing protocols. The framework provides modules for various layers of Sensor Node stack. Applications can be implemented by using these framework modules by sub classing the framework classes [6].

OMNeT++ [7], Objective Modular Network Test-bed in C++ is a public-source, component-based, modular and open-architecture simulation environment with strong GUI support and an embeddable simulation kernel. Its primary application area is the simulation of communication networks, but because of its generic and flexible architecture, it has been successfully used in other areas. The OMNeT++ model consists of hierarchically nested modules. The top-level model is the system model, which encompasses the complete simulation model and is referred to as the “networks”. The system contains sub-modules which themselves may have sub-modules. Thus the modules can be described to any depth of nesting as a result able to describe complex

system models as a combination of a number of simple modules. Modules that contain sub-modules are called compound models. Simple modules contain the algorithms in the modules and form the lowest level of module hierarchy. The user implements the simple modules in C++, using the OMNeT++ simulation class library. Modules communicate by message passing which may be a complex data structure. Modules may send messages directly to their destination or through a series of gates and connections to other modules.

The messages can represent frames or packets in a computer network simulation. The local simulation time advances when the module receives messages from other modules or from the same module as self-messages which is the representation of timers in simulation world. These self-messages are used to schedule events to be executed by itself at a later time. Each of the modules has input and output interfaces called Gates through which message passing between modules is achieved. Messages are sent out through the out-Gate and received through the in-Gate. Connections are created between the submodules or between submodule to compound module depending on the requirement of the system or the topology. The structure of a system maybe represented as shown in Figure 2.1 [8].

As a hierarchal model is followed, the messages typically travel through a series of connections that start and end at simple modules. The description of the topology, the structure and specification of the modules, the Gates and connections are specified through the Network Description Language (NED). NED files are not used directly: they are translated into C++ code by the NEDC compiler, then compiled by the C++ compiler and linked into the simulation executable. The actual behavior of the modules are written in C++ code using the OMNeT++ simulation library and the description of the modules, parameters, gates, connections are specified by the NED language.

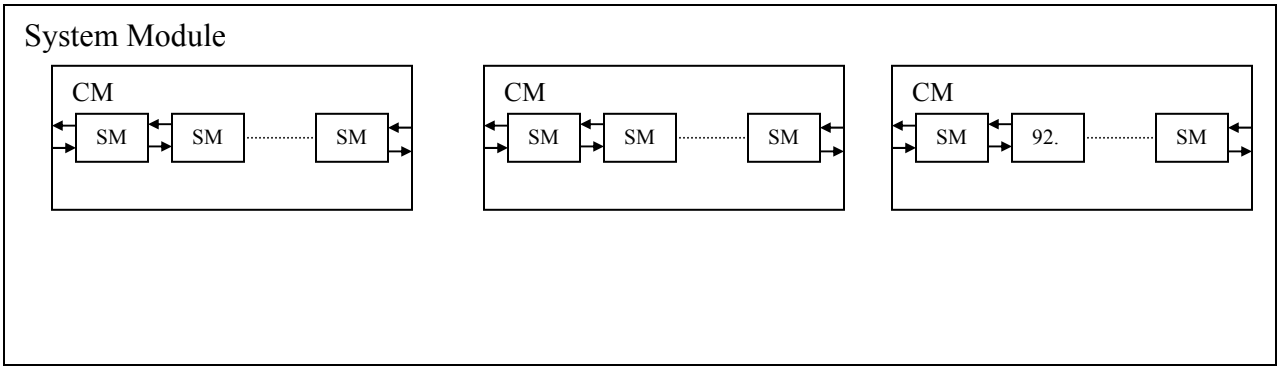

Figure 2.1 Simple and Compound modules in OMNeT++

In this way, there is a separation of behavior and interface definition. This allows reusability of module interfaces defined by NED code. For the implementation of the simple modules OMNeT++ offers an API consisting of a simple module interface, a message interface and a rich simulation library providing support for essential functions, as a lot of routines for the simulation purposes as e.g. I/O-functions, statistics-classes for gathering the achieved results, etc. but also more general stuff like statistical distributions, random numbers generators and even container classes like queues, stacks, containers, etc. Simulations runs are easy to configure and run through initialization files, through which the various data values of the parameters in modules can be specified or changed and simulation re-run with requiring the re-compilation of the simulation setup. In this way OMNeT++ represents a simulation engine, keeping track of the events generated and making sure that messages are delivered to the right modules at the right time, thus accomplishing the task for discrete event simulation.

CHAPTER 3: SENSORSIMULATOR ARCHITECTURE

3.1 SensorSimulator Framework

The goal of the framework provided by Sensor Networking Research Laboratory [8] is to reduce the interdependence between modules and increase the reusability. The development of the simulator was done in such a way that the above goal is achieved. Though many students were working in the project, certain coding practices were followed. The following section gives the architecture of SensorSimulator and a description on the various modules implemented. The architecture closely models a Sensor Network scenario [8] which can be represented by the high-level representation shown below in Figure 3.1. The sensor model can be represented by the Sensor Node model and the Power model.

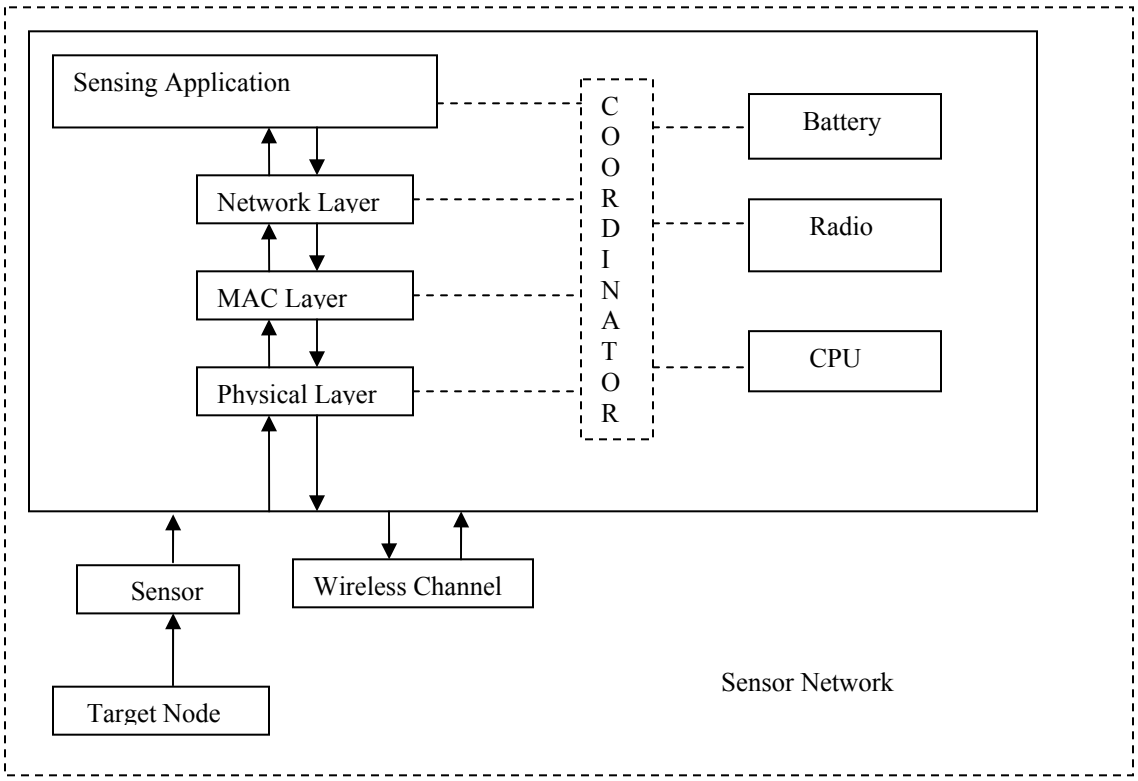

Figure 3.1: Sensor Node Representation in a Network

The framework takes advantage of the design features of OMNeT++. The object – oriented approach makes the framework more flexible. It takes the advantage of modular simulation models as they can be reused flexibly. The Sensor Node model represents the wireless network protocol stack and the sensor applications. The power model represents the hardware of the sensor node: the CPU, Sensor and RF transceiver. The two models act in parallel and achieve the task assigned to a sensor node. The state of the hardware model is changed based on the operation carried out by the software model of a Sensor Node. The power model in a Sensor Node is hardware abstraction of sensor node. This interacts with sensor node model to estimate the power usage. The power model comprises of a single energy source and many consumers. The battery module is a provider with finite amount of energy. The consumers are radio, CPU and other sensing devices that maybe added to the device as illustrated in Figure 3.1. The dashed connections between the CoOrdinator module to all other modules represent the direct communication between it to others. And the arrowed lines represent the gate connections between modules. Consumers report their power usage change to the energy source (battery) and the energy source updates the energy.

SensorSimulator framework consists of the network of sensor nodes that can communicate by wireless means which is supported by the Wireless Channel Module. The layers of network stack in the sensor model are configurable based on the protocol needed for the simulation. The users of this framework have to write the code for their own protocols and integrate it into the framework. The configuration file has all the parameters that are configurable and the simulations are proceeded as desired by the user. The parameters can be changed without any need or recompilation or changes in code.

As the basic functionality of sensors is to sense or detect events or conditions in the surrounding environment, the framework can be described in a target tracking application. The sensing application maybe described as the sensors detect events that are generated by a target or object in the environment near the sensors. The sensors then need to report the event or data collected to the base station or the sink using a multi-hop routing approach in an efficient manner. The sensor network can be represented by sensor nodes that sense and detect events, the Target Node that generates events and the Sink that consumes the data or the final destination of data delivery or the node that can query the network to obtain specific data. The sensors sense events through the sensor channel, which is the representation of the sensing propagation model. The detected events are propagated across the network through the wireless channel that has implementation of the different propagation models in wireless medium [8].

The target node detects the events and sends it to the surrounding nodes that are reachable through Sensing Channel. The application layer of a sensor node receives the message from the Target Node and sends it to other nodes through its protocol stack. The process takes place according to the different protocols implemented at each of the layers. The functionality of the framework and an abstract view of sensor network are described below.

The target node traces a fixed or random path across the network at a configurable speed. It then (target node) sends stimuli through the sensor channel to only those sensor nodes in the vicinity of the target node. A sensor node will be able to receive the stimuli only if the signal strength power of the received packet is above a certain threshold. The propagation model configured at the sensor channel determines the attenuation of the signal and the received signal strength power.

The kind of application demands the implementation of the algorithm or protocol at the Sensing Application layer. The application can be aggregation, cluster functionality, security implementations and other in-network processing implementations.

Hence the sensed data by a node reaches the sink or the base-station through the wireless channel. Since, the transmission region for any typical sensor device is very less compared to the distance that the base station is located, the data has to go through many nodes to the sink and hence a multi-hop route needs to be followed with the other sensor nodes acting as routers to pass on the message to the sink. The energy and memory constraints of a sensor node can make the node fail or die due to power depletion or other environmental conditions and so the network topology will always change. The critical task of the routing protocol is to be able to handle the dynamics of the network and be able to transmit the data to the sink in a reliable, timely and energy efficient or conserving manner. Various routing protocols implemented validate the functionality of the simulation framework. The wireless channel with different propagation models enables the transmission of data between nodes. The propagation models include free space model and the two-ray propagation model.

3.2 Design Approach of the SensorSimulator Framework

The simulator is designed in the form of a layered architecture and the communication between the different layers and modules are accomplished through message passing [6]. The implementation has a hierarchical structure where in the code is divided into base classes and sample classes. Any layer has a base class definition which is a derived class from LayerBase. LayerBase defines the properties of any layer in the

protocol stack. It defines the gate connections and the required parameters that are needed for any layer.

The base class derived from LayerBase has the properties for that particular layer. The sample classes derived from the base class has the implementation needed by users. Hence a user has to derive his class from the base class for his protocol at a particular layer. In this way, all the protocols in samples directory are independent of each other and can also be used collectively. These details are explained in the following section for all the layers. The following section also describes the different modules of the framework.

Common directory has CoOrdinator, packets structures for Network and Mac layer and other constants and attributes used for simulation. This directory is derived by all other directories of the samples directory.

The SensorSimulator simulation is defined in the SensorNetwork module, a compound module that contains all the simple modules, the Target Node, Sensor Nodes, Wireless channel and Sensor channel

3.2.1 Target Node

TargetBase class is the base class that represents the Target Node. It has the base class functionalities that are essential for any target node which includes the position of the target node, its id, its speed etc. TargetNodeSimple extends the TargetBase and has the functional implementation of any target node. Any target node module maintains gate connection with the sensor channel. The simple class generates the stimuli and passes the message to the sensor channel.

3.2.2 Sensor Node

The SensorNode module describes the behavior of a sensor node in the simulation. It is a compound module with different layers of the protocol stack as its sub-modules. The SensorNode module definition and the class represent all the components of the sensor node.

One of the features that the SensorSimulator incorporates in a sensor node module description is the addition of a coordinator module that acts as an interface between all the other sub-modules or layers in a sensor node. As such, the Coordinator Module has the functionalities that coordinate the activities of the hardware and the software modules in a sensor node. The module has to be extended with an added functionality for accessing the properties of the newly added hardware modules or consumers. It has reference to all the layers of a sensor node and all the layers in the sensor node may access the Coordinator. Hence with the help of this coordinator module, any layer may access and update the properties of the other layer. When there is any transmission or reception of a packet, the physical layer has to inform the battery about the decrease in its energy accordingly. This is not done directly between physical layer module and the battery module but is done through the coordinator. Hence physical layer module indicates the coordinator module and this in turn indicates the battery module. The important advantage of this feature is that the individual layers need not have a reference to each of the implementations of a layer. It can have reference only to this coordinator module.

The Coordinator class is responsible for registering the sensor node to the sensor network. It implies that when the coordinator module of a sensor node is initialized, it is given a node id and is also stored in the network level parameter (pNetwork) used in the

simulator). It also implies that the node is up and functioning. The initial energy of the node is taken from the configuration file and when the node is depleted of all the energy, it is unregistered from the sensor network and is considered as dead.

3.2.3 Hardware Model

This forms a very important module for Sensor Network Framework which is of not much importance in other network scenarios. The hardware model comprises of energy consumers and a provider as described in the functional model of a Sensor Node. BatteryBase of the battery module forms the abstract class for different battery models. BatterySimple is a subclass of BatteryBase and updates the energy depending on the number of consumers and the state of activity of the consumers. The specifics of the energy consumption rate and the operation maybe extended to the battery model.

The CPU model has CPUBase that forms the abstract class for the different CPU models. It consists of the basic functionality of a CPU module. CPUSimple, a derived class from CPUBase has implementation of the power consumption of the CPU in different operating modes namely idle, sleep and active state.

Radio Model has RadioBase that acts as an abstract class for the different Radio models. RadioSimple, a subclass of RadioBase updates the energy of the battery depending on the state of the Radio. Radio can be in Idle, Sleep, Transmit and Receive state. The configurable values for the different properties of the hardware and consumers are provided through the configuration file.

The software model represents the different layers of the wireless protocol stack: The implementation at the Application Layer consists of the application specific functions and other in-network processing such as aggregation and passing the result on to the network layer. The Sensing Application Layer can receive information from the

Sensing device of the Target Node through the sensor channel and take the appropriate action based on the application.

The Network Layer implements the routing protocol for sensor networks which forms a very important task of information processing. The Simulator has various implementations at this layer. Random Asynchronous Wakeup protocol for Sensor Networks, Optimized Broadcast Protocols for Sensor Networks, Efficient Coordination Protocol for Wireless Sensor Networks, Directed Diffusion with Geographic aware routing protocol, Max-Min Length-Energy-Constrained Routing, Energy Aware Routing [9] have been implemented in this layer. The network Layer receives the message from the application layer, and then transforms the message to a macPacket type message and sends it to the bottom layer to the MAC layer. The NetworkPacket maybe broadcast or unicast to specific node (sink node).

The MAC Layer has MAC_802_11 (IEEE 802.11 with Distributed Coordination Function implementation) and Simple Mac implementation. The messages received form the MAC module are encapsulated at the physical module and are sent to the WirelessChannel Module. The physical module interacts with the radio model through CoOrdinator and transforms the state of the radio before sending the message to the WirelessChannel Module. The Wireless Channel receives the message through one of its multiple gates. Energy is updated at regular intervals in the Battery Nodule as and when the state of the consumers changes.

The WirelessChannel Module represents the medium through which the sensor nodes communicate. Any message from a node to the wireless channel is sent to all the neighbors within its transmission region with a delay d, where d is (Distance between the communicating Sensor Nodes) / Speed of Light. The message from the physical layer

consists of node id and other parameters needed by the wireless channel for transmitting it to the neighbors of that node.

Radio Propagation models implemented at the Wireless Channel Simple are used to predict the received signal power of each packet and these affect the communicating region between any 2 nodes. The Radio Propagation models are derived by the WirelessChannel Module.

Free Space Radio Propagation Model assumes the ideal propagation condition that there is only one clear line-of-sight path between the transmitter and receiver.

Two-ray ground reflection model considers both the direct path and a ground reflection path as a means of communication between two mobile nodes. This model gives more accurate prediction at a long distance than the free space model.

3.3 New Implementation

Every layer has three basic files: .cc,.h and .ned files. User has to have all these three files in a layer for his implementation. We are providing these three files for every layer with minimum functionality. User can add his code and just do “make” and run the simulation by changing the necessary configurable parameters in the omnetpp.ini file in the samples directory [6].

Move into samples directory. This directory has sub-directories each for a layer. Each directory has examples of various implementations at each layer. Also, you can see the files New* Layer.cc, New*Layer.h and New*LayerModuleDefn.ned for a new user to start using the simulator. You can add the various parameters of a module which you will be using in your implementation in the .ned file.

After adding the code or making changes for the existing implementation, do make and then go back to samples directory. The class name of this new implementation has to be specified in omnetpp.ini so that the simulation considers your implementation for execution. Eg: sensorNetwork.Nodes[*].strMACLayerType="MAC_802_11"
With this functionality, user can just add his protocol at that particular layer with all other protocol layers being same.

CHAPTER 4: COMPARATIVE STUDY OF DIRECTED DIFFUSION

The initial version of SensorSimulator is verified by making comparisons with ns2. The performance of the simulator is tested in terms of execution speed and memory consumption. This is done by implementing Directed Diffusion routing protocol with a similar setup as that of ns2. The implementation details are maintained same in both of them. The reason to choose Directed Diffusion was that it being a very well known and useful protocol implementation for a new simulator. The comparisons with a protocol that is under research will help further study on it.

Directed Diffusion is a new data dissemination paradigm for sensor networks and is data centric. Data generated by the nodes have attribute value pairs. Interests are generated by nodes for the named data and the data matching the interest is drawn down towards that node [15]. Intermediate nodes cache, or transform the data.

The details of the protocol are as follows: The nodes that generate queries are called subscriber nodes and generate queries at a regular interval. The subscriber node initially generates beacon messages and get the information about neighbors from beacon replies and then forwards the query. Each node follows this procedure and use Geographic Routing to forward the query to the region. If a node in the path does not have any neighbors or all its neighbors are away from the region, then it sends a message to its parent node that it is a dead-end. The parent node on updating the cost of the unreachable node, forwards the query in an alternate route towards the region. In the specified region, the interest is recursively flooded. The interest cache is maintained at each of the nodes in the path with its gradient of interest to each of the neighbors. The nodes in the region that have the specified properties of the interest send out data. These nodes are referred to as Publishers [8].

The data is marked as Exploratory to reinforce the path that was taken by the interest. On receiving the data marked as Exploratory by the subscriber, positive reinforcement message is sent out by the Subscriber node. Each node on path forwards this message thus reinforcing the path to the region. When the node reinforces a path, its cost to the region is known and this cost is sent back to its source node, which updates the cost information of that node to the particular region of interest. Thus the path with the highest cost is always maintained, reinforcing the route. The data from the region follow the path established by the reinforced messages. The nodes in the region send out data at the rate that is specified in the query. Data caching is implemented in intermediate nodes and so the data requested by different subscribers from the same region maybe satisfied by the common node in the path thus reducing the traffic and redundant messages. The data marked as exploratory are sent to identify better paths and reinforce at regular intervals. Also the neighbor- updating procedure phase is carried out, i.e. at regular intervals the beacon messages are broadcast and beacon-reply messages are sent by neighbors thus maintaining latest neighbor information [8].

The analysis initially focuses on the validation of the Directed Diffusion implementation. In this experiment, the set up had a Sensor Network with different number of nodes between 5 and 200. For each Sensor Network, we identified the maximum size of the sensor field (with respect to grid coordinates). A fixed number of query generating nodes are distributed randomly in the sensor field. Next, the target region with a specified boundary in terms of the grid coordinate is selected. The region has a certain number of sensor nodes to send the data back to the query generating nodes. The simulations were executed for a certain time duration and the ratio between the number of packets generated in the region and the number of packets received by the

query generating nodes are observed. 802-11 MAC is being considered at the MAC layer with a simple pass through Physical layer for these simulations in both of them. The results for 5-200 nodes are shown in Figure 4.1 for various topologies. The results show that on an average, for a similar topology and simulation environment, the delivery ratio is comparable with ns2. This validates the implementation details of SensorSimulator with that of ns2.

Another series of experiments use MAC 802.11b at the MAC Layer and test the execution speed of SensorSimulator with ns2. A simple pass through Physical Layer is considered. The simulations are executed for 100, 500, 1000 and 2000 nodes. The nodes in the Sensor Network are deployed randomly in various locations with the network size configured such that the node density is maintained constant. The simulation is setup for 10 nodes generating queries and 10 nodes in the region and is run for a period of 300 simulation seconds. The simulation is setup in the two simulators such that the nodes have the same coordinates, deployed in the same network size with publishers and subscribers generating queries and data at the same interval. This is to make sure that the simulation setup is the same in both the simulators and then the performance is tested. The performance of the simulation is as shown in Figure 4.2. The comparison shows that, though initially the behavior of both the simulators is similar, the deviation is seen when the number of nodes are increased. The performance graph of ns2 deteriorates for nodes greater than 500. And when the number of nodes is increased to 5000 the simulations hanged and the statistics could not be collected. The graph of the SensorSimulator is linear when the nodes are increased from 500 to 2000.

It is also observed that the memory consumption for 10 queries in SensorSimulator is very less compared to ns2. The memory consumption Figure 4.3

shows that the data structures used for the simulation are used in a scalable manner to represent the different classes and the interaction with the framework. It can also be observed that the rate of memory usage increases at a faster rate for ns2 than for SensorSimulator thus allowing for large simulation setup and ability to simulate larger, scalable networks.

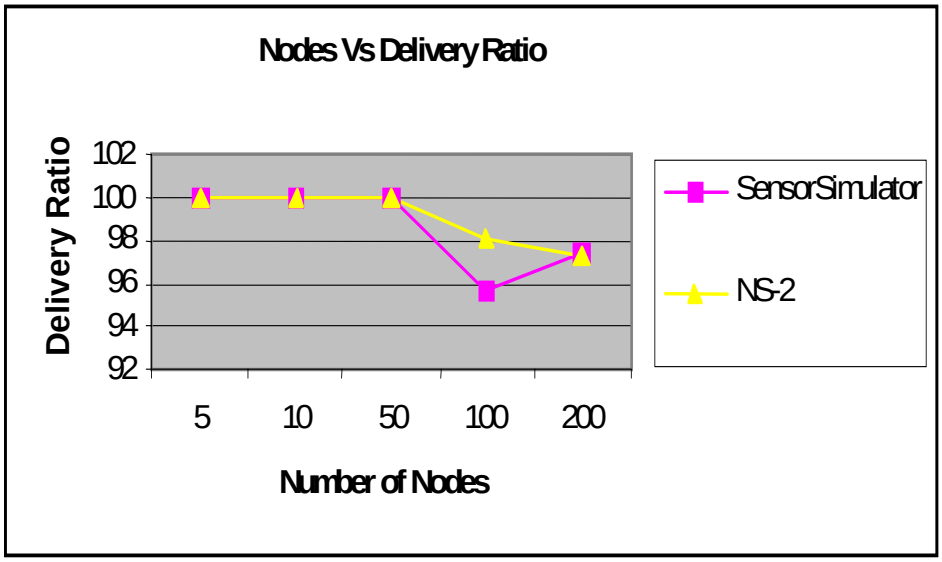

Figure 4.1 Comparison of Delivery Ratio

With the following comparisons, we would like to develop the library of protocols supported by the SensorSimulator. Since a very good simulator should be able to support many protocols, we included Min-Max Energy Constrained Routing Protocol developed by the Sensor Networking Research Laboratory at LSU and a good broadcasting protocol that will be discussed in the following section.

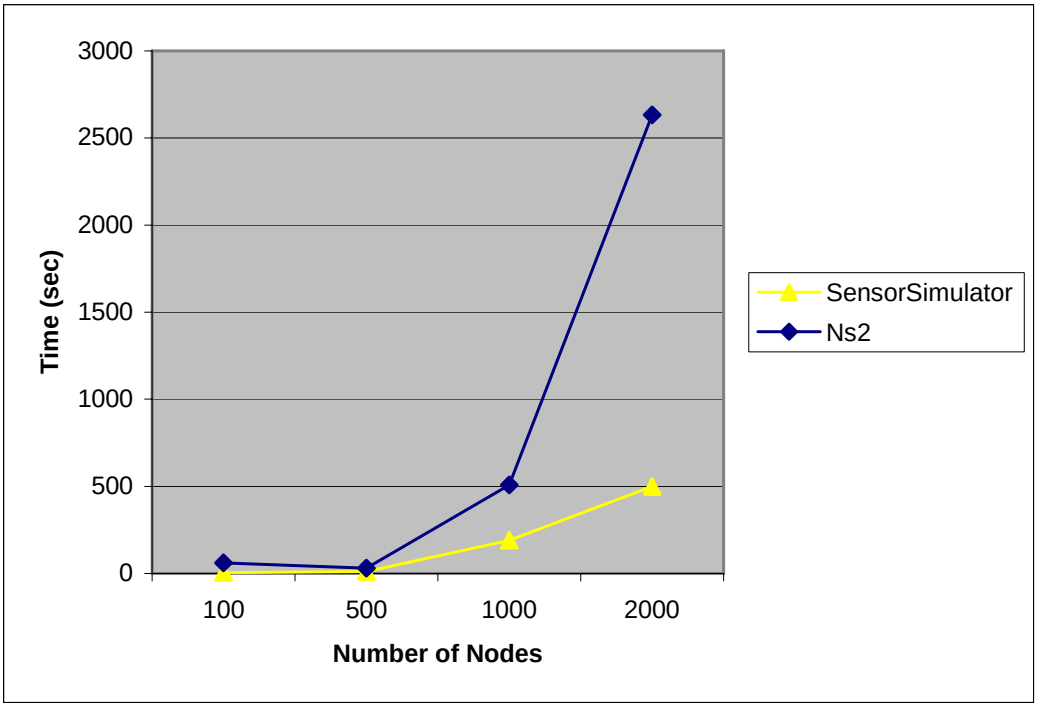

Figure 4.2 Execution Time Comparison – Directed Diffusion Vs MAC802.11

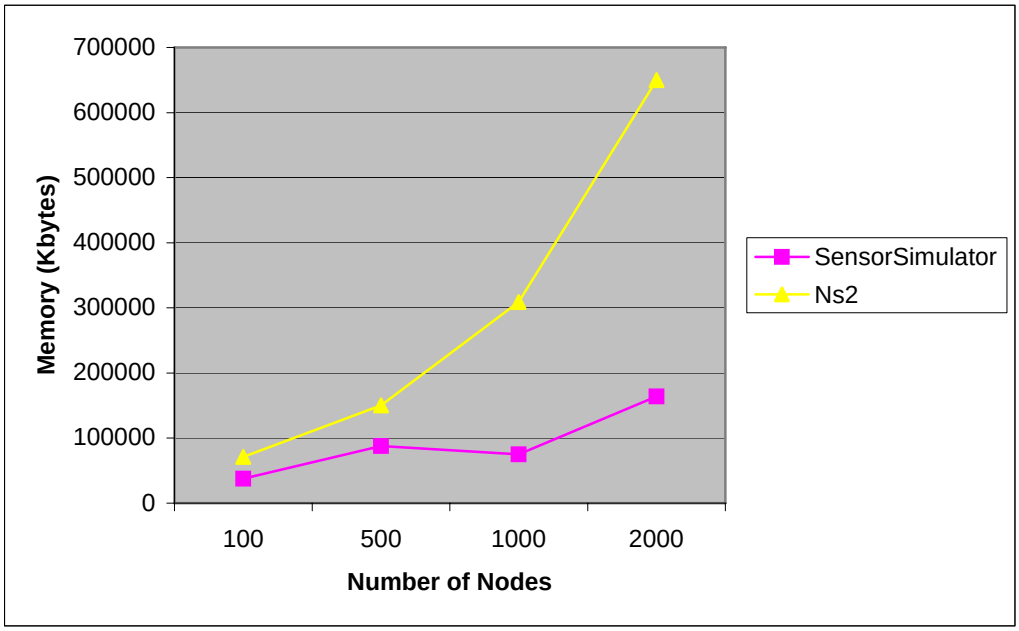

Figure 4.3 Memory Usage 10 queries

CHAPTER 5: IMPLEMENTATION OF BPS

5.1 Study of Routing Protocols in Wireless Sensor Networks

The application areas of Distributed Sensor Networks necessitate spreading the sensor node devices in a sensor field. Each of them has the capability to collect the information and route it back to the user. Data is routed by a multihop infrastructureless architecture. The design of a sensor network is influenced by many factors such as fault tolerance, scalability, operating environment, sensor network topology, hardware constraints, power consumption etc. These factors serve as a guideline to design a protocol or an algorithm for a sensor network. The protocols that are discussed in the following sections consider most of the factors stated above for their implementation.

The current thesis describes the simulation study of different routing protocols Routing Protocols in Wireless Sensor Networks.

One of the important tasks for a sensor node is to save its energy and maintain the lifetime of a sensor network. Hence energy efficient communication techniques are very essential since battery power is a scarce and expensive resource in the devices. Broadcasting is a very energy-expensive protocol and is also widely used as a building block for other network layer protocols. Therefore, reducing the energy consumption by optimizing broadcasting is a major improvement in sensor networking. And the implementation of an optimized broadcasting protocol in the SensorSimulator enables various other network layer implementations in the simulator.

5.2 BPS

The proposed optimized Broadcast Protocol for Sensor Networks (BPS) [10] uses adaptive-geometric approach that enables considerable reduction of retransmissions by maximizing each hop length. In BPS, nodes do not need any neighborhood information

and this leads to low communication and memory overhead. In the worst case scenario, the number of transmissions is a constant multiple of those required in the ideal case. Our simulation results show that BPS is very scalable with respect to network density. It is also resilient to transmission errors. Any network layer should consider the design factors as discussed above and hence BPS considers the following goals for its design: Scalability, Energy efficiency, memory and computation [10].

The number of transmissions needed for broadcasting are minimized by doing selective forwarding, where only a few self-selected nodes in the network retransmit packets and this is done by maximizing each hop length. That is, after one node has transmitted, it would be desirable that the next transmitting node to be the most distant possible. BPS uses a self-selection mechanism to decide which one will transmit next. So, after one node has transmitted, the next retransmitting node will be self-selected being the most distant (bounded by communication range) one from the source that was able to receive (construct) the packet without errors. This is achieved by using a waiting mechanism that imposes to nodes a waiting time inverse proportional to the distance from the source.

The key advantages of BPS are

a) By minimizing the number of unnecessary transmissions and by maximizing the hop length, it outperforms other variations of flooding

b) The radio channel quality is measured implicitly for each transmission and hence the protocol adapts itself and tries to get the best out of the existing radio channel conditions.

c) The number of transmissions required decreases as the density of the network increases.

d) A node need not know the locations/addresses of all its neighbors and hence BPS does not impose any bandwidth overhead in terms of *hello* messages and has no memory overhead

e) The protocol performs well even in very large networks

All these factors make the protocol well suited for sensor networks that operate even in adverse conditions [10].

5.3 BPS Algorithm

The algorithm initially describes the ideal case scenario. The area to be covered with radio signal is portioned into hexagons. The length of the hexagon side is determined by the communication range of a node. The Source is at the center of one of the hexagons. In an ideal network, all other transmission nodes are at the vertexes of the hexagons and are called the strategic locations. The strategy to select such nodes was inspired by the Covering Problem addressed by Kershner [16], that no arrangement of circles could cover a 2-dimensional plane more efficiently than the hexagonal lattice arrangement. The broadcasted packets are propagated along the sides of the hexagons (except the first round of transmissions). Any node located inside a hexagon is reachable from at least one of the vertex nodes of the hexagon [10].

In a real scenario, it is impractical to assume that the nodes are located at the vertexes of the hexagons. Hence if the neighbor nodes are not in the optimal strategy locations, the coverage figure will be distorted and the distortion effect may propagate. A simple solution is to select the nearest node to the supposed vertex.

It can also be observed that a node can receive a packet more than once, from different directions and from different nodes, each node specifying different optimal strategic location. This may cause two nodes very close to each other to retransmit. We

propose to avoid these transmissions by having a node keep track of its distance "dm" to the nearest node that has retransmitted the packet and to have a node retransmit only when its distance to the nearest transmitting node is greater than a threshold "Th". Each broadcast packet contains two location fields, L_1 and L_2 in its header. Whenever a node transmits a broadcast packet, it sets L_1 to the location of the node from which it received the packet and sets L_2 to its own location.

The algorithm is as follows [10]: The Source Node *Src* sets both *Loc1* and *Loc2* to its location (S_X, S_Y) and transmits the packet.

a) Upon the reception of a broadcast packet, a node *N*, first determines if the packet can be discarded. A packet can be discarded under any of the following conditions: If the node has transmitted the packet earlier. If a node which is very close has already transmitted this packet, i.e., if dm < Th.

b) If the packet is not discarded, *N* determines if the received packet comes directly from the broadcast source *Src*. If yes, *N* finds the nearest vertex V of a hexagon with (S_X, S_Y) as its center coordinates and with (S_X+*R*, S_Y) as one of its vertexes. It computes its distance *l* from V and then delays the packet rebroadcast by a delay *d* given by $d = l/R$. Else, if *N* hasn't received the packet directly from the source Src, but from some other node K, then *N* selects nearest strategic location (hexagon vertex). The packet transmission is delayed by $d = l/20*R$.

c) After the delay *d* elapses, *N* again determines if it has received the same packet again and if the packet can be discarded (for the same reasons mentioned above). Thus, delaying enables the selection of a node that received successfully the packet and is the closest to the corresponding strategic location. In the case that the packet cannot be

discarded, *N* updates *Loc1* to location of the node from which it received the packet and *Loc2* to its location, sets *dm* to zero and transmits.

BPS minimizes the number of transmissions by maximizing each hop length. In ideal conditions the hop length is equal to communication range of nodes. But in real conditions, as shown in several studies [16, 17], there is no clear correlation between the packet delivery and the distance among nodes and there is a significant *gray area* within communication range of nodes, where receivers experience variable and unstable reception over time. Even in this conditions, BPS tries to maximize the hop length by selecting among the nodes that received correctly the packet this time (even in the *gray area*) the one that is the closest to its corresponding strategic location (and more distant from the previous transmitting node). This is achieved by having nodes that received correctly to self delay the retransmission, where this delay is proportional to the corresponding strategic location. The extra transmissions could not be avoided totally.

Low delay values decrease the time needed to broadcast a message all over the network, and a high delay values help reduce redundant transmissions in instances where two nodes are of about same distance from the strategic location. This value lies around 10 ms for the simulations. The delay values are much less than 10 ms for dense networks.

The Protocol implementation in SensorSimulator is as follows:

The Module for this protocol is a derived class from NetLayerBase of the Simulator. Hence it incorporates the basic features defined at the network layer for Simulator. Any network layer message needs to have standard parameters that are needed for MAC layer. Hence the network packet has 2 headers. One header for the standard network message and the other that is specific for this broadcasting protocol. As

mentioned in the algorithm, header has fields specifying the locations of the strategic locations.

The structure of the common header is already specified in the architecture of SensorSimulator. The structure of this protocol header has fields for hexagon locations which specify the stragetic locations for any transmission and the variables that are needed for the purpose of implementation such as Id of the broadcast packet. The configurable parameters include the Threshold Th, and the dn. The node that wants to start the broadcast can also be specified in this header. Once it is selected, a self-message is schedule to create an OFP Packet which consists of calculating the hexagon locations and calculating the nodes nearer to these locations (as assumed that the node positions are stored in a global directory for simulation purpose). These are stored in the header field and are broadcasted. The nodes whose Ids are specified in the header field initially check whether they received a packet with that Id from that source. If not they set the field to retransmit the packet, set their dn to zero and set the retransmission flag. All the other nodes update their dn value by calculating the distance between them to the node from which they got this packet. The retransmitting nodes in turn select the next retransmitting nodes as specified by the algorithm.

A wireless network of different physical areas having different number of nodes was simulated. To be more specific, circular regions of radius varying from R to 10R and rectangular/square regions of size varying from 3R X 3R to 10R X 10R have been simulated, where R is the communication range of each node, which was set to 300m in all our simulations.

The nodes were uniformly distributed all over the region with the density varying from 4 nodes per R X R region to 100 nodes per R X R region.

The simulations study the performance of BPS in networks of different sizes and densities. Initially, we studied the effect of different threshold values on the performance of BPS. Then, we were concentrated on the algorithm efficiency by studying the performance of BPS in static networks and also in highly mobile networks. Lastly, we studied the performance of OBS in networks where the coverage area of a node is not circular. The simulation results under each network study are presented in a subsection below.

5.4 Effect of Threshold

This experiment evaluates the effect of different threshold values on the performance of BPS. Table 5.1 shows the simulation results for threshold values of 0.35*R, 0.4*R and 0.45*R. Apart from the number of transmissions in each case, the delivery ratio in percentage for each case is indicated at each data point. Delivery Ratio is the ratio of average number of nodes that receive the message to the total number of nodes in the network.

For a threshold value of Th = 0.35*R, a delivery ratio of around 98% is achieved and for Th = 0.4*R, the delivery ratio is close to 95%. But, for Th = 0.45*R, the delivery ratio falls to around 90%. This is understandable, because with the increase in threshold value, the number of retransmitting nodes decreases [10].

All the further simulations use a threshold value of Th = 0.4*R and the minimum and maximum delivery ratio are shown.

5.5 BPS Efficiency

In this Section we evaluate the performance of BPS in networks of different sizes and different densities. We include a “best-case” bound provided by the simulation results

Table: 5.1 Effect of *Th* on the performance of BPS. Network size is 4R X 4R.

Threshold/Density	4 per RXR	6.25 per RXR	11 per RXR
0.35R	23.1	24	23
0.4R	19.6	18.6	17.8
0.45R	15.5	15.9	15

in ideal case scenarios. And a worst case scenario is considered as a simple flooding. Thus, these bounds provide a useful spectrum to gauge the performance of our protocol. The protocol when compared to simple flooding, uses up to 65% to 90% fewer messages depending on the density of the network. The network size is varied from 3R X 3R to 10R X 10R, while keeping the communication range of each node fixed to 300m. We also varied the network density from four-nodes/R X R region to 100-nodes/R X R region. Figure 5.1 is a plot between the number of transmissions required to cover entire region for varying densities and for different areas of the region [10]. Network areas up to 10R X 10R have been considered. Figure 5.2 presents the results in a different perspective [10]. It gives a plot between the number of transmissions and density of the network for different network sizes. It can be seen that the number of transmissions required decreases as the number of nodes (density) increases. The number of transmissions at a density of 100 is very near to the number needed in an *Ideal case*. The minimum delivery ratio achieved by BPS was 94.3% for the case with network size of 6R X 8R and with a density of 6.25. In all other cases, the delivery ratio was close to 95% with the maximum being 97.3%. The results show that the performance of BPS remains very efficient even in large networks; network size does not seem to affect the performance of BPS.

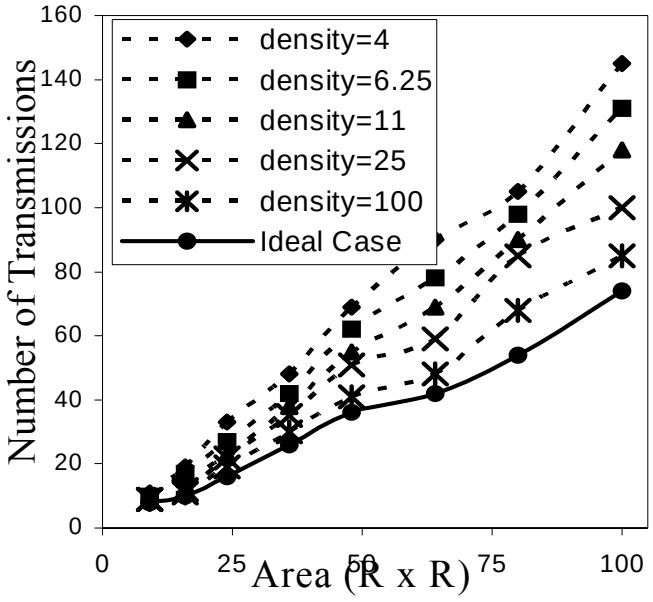

Figure 5.1 Number of transmissions required to cover an entire region for different areas

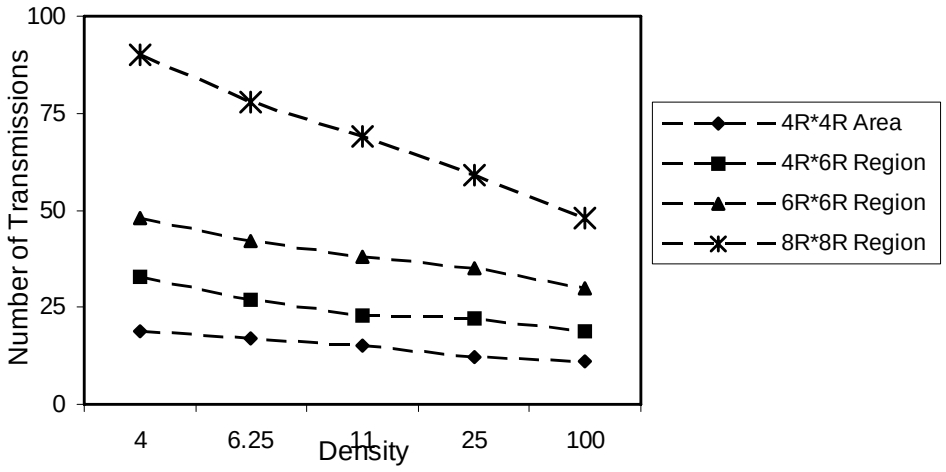

Figure 5.2 Number of transmissions for varying node densities for different areas

CHAPTER 6: IMPLEMENTATION OF ECP

The Wireless Channel is usually shared among many nodes in a Sensor Network. This sharing increases the complexity of route discovery, reduces the network performance, and increases energy consumption due to aggravated radio interference. Topology control is a technique used in sensor networks to address these problems. Topology control optimizes network topology and reduces routing cost by restricting the connections among pairs of hosts. Another important application of topology control is to save energy in sensor networks.

One approach of topology control is to exploit the node redundancy in sensor networks. Sensor networks have high level of node redundancy because of the high density. Each node can reach a number of neighboring nodes. Therefore a subset of nodes can be selected to serve as the coordinators through which all nodes can, directly or indirectly, communicate with each other.

The coordinators form the backbone of the network. The nodes that are not in the backbone have at least one neighboring node that is in the backbone, i.e. all the nodes in the network are connected through the backbone. The non-backbone nodes that do not have active communication can safely go to sleep to save energy. The duration of sleep time depends on how long the backbone can be maintained – which is usually dozens of seconds.

In sensor networks, node unreliability and high cost of transferring information across the whole network make it impractical to use a centralized backbone algorithm. Thus, many distributed algorithms have been proposed.

Another characteristic is identified for wireless sensor networks. Nodes in the connected dominating set consume more energy to handle various bypass traffic than

nodes outside the set. Therefore, static selection of dominating nodes will result in a shorter life span for certain nodes, which in turn results in a shorter life span of the whole network. To prolong the life span of each node and, hence, the network by balancing the energy consumption in the system, nodes should be alternately chosen to form a connected dominating set. In an ad hoc network, the random mobility of nodes might implicitly achieve a protocol to achieve this objective, but in a sensor network, where sensors are static, the protocol needs to explicitly provide balancing of energy among nodes [21].

An Efficient Coordination Protocol (ECP) is proposed to address the above issues. We propose to save overall energy consumption by selecting a few sensors that form a connected dominating backbone and keeping them awake [21]. Selection of dominating backbone network is based on the extended Covering Problem [10]. We allow only dominating nodes (i.e. backbone nodes) to participate in routing. In case of a broadcast message, only the backbone nodes retransmit the broadcast packet, to reduce broadcast redundancy. In addition, in order to maximize the life span of all nodes, ECP periodically selects nearly disjoint subset of nodes to form a connected dominating set.

ECP does not require any neighborhood information and consequently scales well with the number of nodes and network size. The effectiveness of the protocol is studied through simulations carried out in SensorSimulator. The simulation results also show that system lifetime with ECP is significantly better than without ECP, for a range of node densities, without much reduction in overall forwarding capacity.

6.1 Problem Statement

We study the problem of designing an efficient and distributed algorithm that partitions the sensors in a WSN into k covers such that each cover forms a connected

dominating set, thus yielding a virtual backbone. The problem of choosing which cover a sensor will belong to is abstracted into SET-K-CDS problem and can be stated as follows [21]:

6.1.1 Problem SET-K-CDS

Given: A graph G (V, E)

To find: A partition ζ of the graph into *k* subsets S_1, S_2… S_k, where each subset is a dominating set. Criteria is to Maximize |S| and Minimize $\mu = \sum_{\substack{\forall i,j \\ i \neq j}} | DS_i \cap DS_j |$.

Informally, we are maximizing the number of dominating sets while making the dominating sets themselves as independent as possible. Ideally, there should be no overlapping between any two dominating sets (μ = 0). This implies that each node belongs to one and only one dominating set. Also, it is desired that all nodes are part of some dominating set, as this results in perfect load balancing. Additional criteria can be added. For instance it might be preferable to have all sets to be of uniform size. Once the partition is obtained, in an effort to increase the longevity of the network and conserve battery power, it would be beneficial to activate groups of sensors in rounds, so that the battery life of a sensor is prolonged and the same time connectivity is maintained. Rather than using all the sensors all the time to forward packets and maintain connectivity for events, SET-K-CDS solutions provide a simple way for sensors to share in the tasks, so that their energy resources can be conserved [21].

Computing an optimal partition ζ meeting the above criteria is NP-hard. Also, achieving the complete independence among dominating sets might be too strict and impractical. The proposed ECP balances the load among the sensors to a great extent.

6.2 Protocol Description

ECP adaptively elects Backbone Nodes (BNs) from all nodes in the network. BNs stay awake continuously and perform multi-hop packet routing within the sensor network, while other nodes remain in power saving mode and periodically check if they should wake up and become a BN.

The protocol achieves following objectives: First, it ensures that enough BNs are elected so that every node is the radio range of at least one BN. Second, it rotates the BNs to ensure that all sensors share the task of providing global connectivity roughly equally. Third, it attempts to minimize the number of nodes elected as BNs, thereby decreasing network energy consumption and thus increasing network lifetime, but without suffering a significant loss of capacity or an increase in latency.

We assume that each node knows its location which itself is a requirement for various routing protocols, sensing, target tracking and other applications. Various techniques like GPS [11] have been proposed to enable a node to discern its relative location. We also assume that the nodes are loosely synchronized.

The protocol runs above the MAC layers and interacts with the routing protocol. ECP leverages a feature of modern power-saving MAC layers, in which if a node has been asleep a while, packets destined for it are buffered at the forwarding BN. When the node awakens, it can retrieve these packets from the buffering BN. ECP also requires a modification to the route look up process at each node – at any times, only those entries in a node's routing table that correspond to currently active Backbone Nodes can be used as valid next hops (unless the next hop is the destination itself).

The following sections describe how a node decides it should be a BN.

Periodically, base station initiates the backbone reconfiguration procedure. The solution is based on the extended covering problem. The objective is to select sensors in the network that would form the best approximate for a hexagonal lattice structure to cover the whole area. The algorithm is similar to the one (BPS) discussed in the previous section.

The Initiator constructs a *BN_formation* message with two location fields L_1 and L_2 in the header. Whenever a node transmits a broadcast message, it sets L_1 to the location of the node from which it received the message and sets L_2 to its own location.

The protocol is as follows:

The Initiator *S* sets both L1 and L2 to its location (S_X, S_Y) and transmits the message. A node M, upon receiving a BN_formation message, first determines if the message can be discarded. A message can be discarded, if the node has transmitted the message earlier or if there is a Backbone Node (BN) closer to it by a distance less than *Th,* where *Th* is a threshold.

If the message isn't discarded, M determines if it received the message directly from the initiator S. If yes, M finds the nearest vertex V of a hexagon with (S_X, S_Y) as its center and with (S_X+R, S_Y) as one of its vertices. It computes its distance *l* from V and then delays the message rebroadcast by a delay *d* given by $d = l/R$. Else, if M hasn't received the message directly from the source S, but from some other node K, then using properties 1, 2 and 3 mentioned in section 3 and with the nearest strategic location. The message transmission is delayed by $d = l/20*R$.

After delay *d*, M again determines if it has received the same message again and if the message can be discarded (for the same reasons mentioned above). Thus, delaying enables a node to decide if it is the nearest node to the strategic location.

If the message cannot be discarded, M advertises itself as a Backbone Node, updates L1 to location of the node from which it received the message and L2 to its own location and transmits the message.

The above protocol itself might not guarantee that all sensors in the network are within one-hop distance from some BN. To overcome this, we make the following extension to the protocol:

Each sensor that does not have at least one BN in its neighborhood sends *BN_enquiry* message to all its neighbors. Each sensor N that already has a BN as its neighbor, stores the information of all distinct *BN_enquiry* messages it receives. N sets a timer that is inversely proportional to number of *BN_enquiry* messages it received. If it receives no *BN_update* before the timer expires, it advertises itself as a BN and sends a *BN_update*. The *BN_update* also consists of list of all *BN_enquiry* messages it received. This helps other sensors update their BN_enquiry list and reset their timer. This process continues until all sensors are within one-hop distance of some BN.

Optimal Backbone network resolves contention by delaying BN announcements with a back-off delay. Each node chooses a delay value, delays the *BN_announcement* message that announces the node becoming a BN. We consider a variety of factors in our derivation of the back off delay. Consider first the case when all nodes have roughly equal energy, which implies that only topology should play a role in deciding which nodes become coordinators.

6.3 Performance Evaluation

The performance of ECP is tested on SensorSimulator and the protocol is compared with blind flooding. Wireless networks of different physical areas with different number of nodes were simulated. To be more specific, square regions of size

varying from 3 X 3 to 10 X 10 have been simulated, where the transmission range of each node is considered as one unit. The nodes were uniformly distributed all over the region with the density varying from 6.25 nodes per unit area to 25 nodes per unit area.

Initially, we study the size of the backbone network and study the performance of ECP with respect to the delay and energy savings. Then we present the performance in terms of balancing the energy.

Figure 6.1 shows the size of backbone network for different network sizes for different densities. The graph shows that the size of the backbone network scales with density and is fairly constant for a given network area.

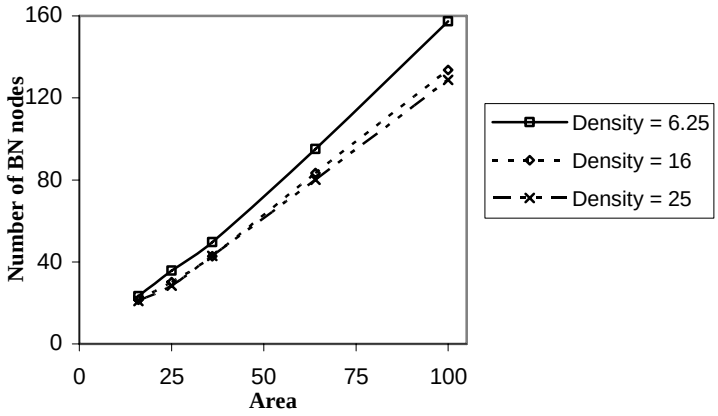

Figure 6.1 Number of Backbone Nodes for different network sizes and densities

Figure 6.2 gives the end-to-end packet delay for different loads. In this case, each node generates packets at a given rate. If a node is asleep, whenever it has a packet to send it transmits the packet to one of the backbone nodes that is nearest to the destination and then goes to sleep again. We compare the delay in ECP with that of RAW. It can be seen that, the delay with ECP is marginally greater than delay with RAW. This can be attributed to the fact that each packet might traverse longer hops with ECP when

compared to the global shortest path in case of RAW where all the nodes are active all the time.

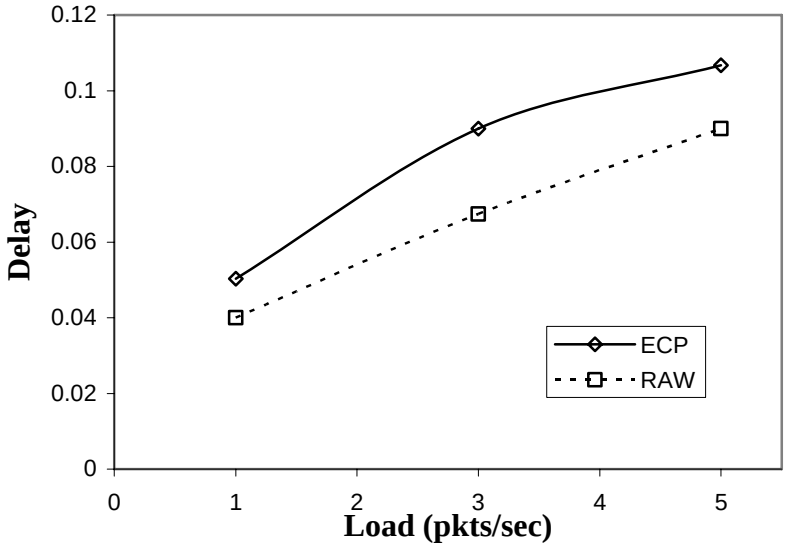

Fig 6.2 End-to-end packet Delay

CHAPTER 7: IMPLEMENTATION OF RAW

As discussed in the introduction Chapter that sensor networks are energy constraint, energy efficient communication techniques are very critical for increasing the lifetime of sensor nodes. Hence the design of a good power management protocol for wireless sensor networks needs to consider the following attributes; energy efficiency and scalability to the change in network size, node density and topology whereas latency, fairness and bandwidth, which are generally the primary concerns in traditional wireless voice and data networks, are secondary in sensor networks.

The Random Asynchronous Wakeup (RAW), a power management scheme is explicitly designed for wireless sensor networks [18], focusing on the above discussed issues. With reduced energy consumption, the protocol achieves good scalability and low latency. This is achieved by reducing idle listening; by making the sensors operate at very low-duty cycle modes. And a low duty cycle increases latency and reduces throughput. RAW uses the concept of Stateless Nondeterministic Geographic Forwarding (SNGF) [19].

The Protocol consists of two schemes; routing based on forwarding sets and random wakeup scheme. The routing protocol is designed to take advantage of the fact that sensor networks are densely deployed. Unlike conventional routing protocols, a packet can be forwarded to any of the several paths existing between the source and destination nodes, where the path lengths of these paths are comparable to the shortest path. The Random Wakeup Scheme allows a node to be active during a randomly chosen fixed interval in each time frame.

7.1 The Protocol

Each sensor gets up periodically, transmits a beacon message indicating that it is ready to receive or forward a message. It waits for duration t_x for a reply. If it gets an RTS from any of its neighbor in that duration, it receives the message and extends the duration of its idle time. Then it checks if it can forward the message to any of its neighbor. If no neighbor in the forwarding set is awake it waits until a neighbor is awake. Then it forwards the message to that node and goes to sleep again.

Time axis is divided into fixed-length time frames of length *T*. Let T_{setup} be the time taken by a sensor to send a beacon message once it is awake and receive a reply consisting of it neighbor information. t_x is the duration that the sensor waits for an RTS.

7.2 Description

In order to better explain the protocol, we make use of the state-transition diagram shown in Figure 7.1

7.2.1 Sleep State

The node is in the sleep mode. Also, in order to conserve energy, it is desirable to maximize the time a node spends in sleep mode. Each sensor *i* selects its own schedule period T_i based on different parameters (described below) and sleeps for a duration of T_i'= random(0, T_i) before it wakes up again.

Thus, we propose to set T_i as follows:

$$T_i = T.\frac{Energy_i}{Avg\_energy_i}$$

T is the duration value in case when all sensors have same energy levels. $Energy_i$ is the energy level of sensor *i*. Avg_Energy_i is the average energy level of the neighbors of sensor *i*.

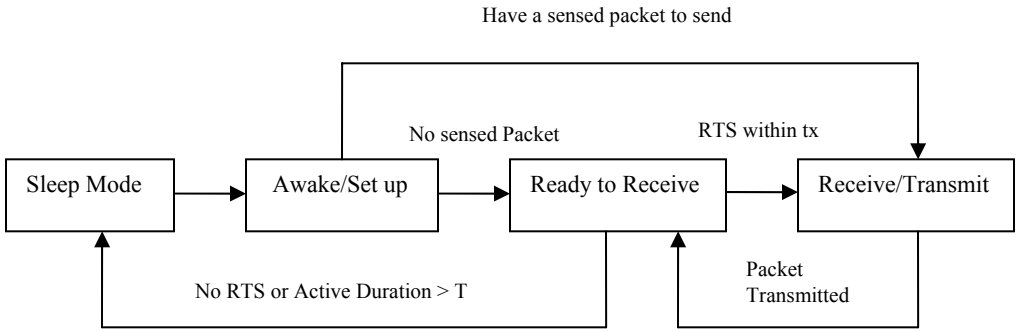

Figure 7.1 State Transition Diagram

In case, when average energy level of neighbors is needed, a node includes its energy level in the beacon message it broadcasts.

7.2.2 Awake / Setup

When ever a node becomes active, it broadcasts a beacon message through the control channel, advertising to its neighbors that it is awake. Then, it checks if there is any packet generated by it to be transmitted. In that case, the sensor goes directly to the Receive/Transmit state, else it goes to the Ready to Receive state.

7.2.3 Ready to Receive state

A sensor *i* once in this state, keeps its receiver antenna active and listens to channel to see if any of its neighboring sensors are forwarding a packet. If it receives an RTS addressed to it, it goes to Receive/Transmit state. If the node does not receive any RTS with in t_x it goes to sleep. Also, to avoid excessive drainage of energy at sensor *i*, we put a constraint that once a sensor is continuously active for a duration more than T_i, it stops receiving packets and just send all the packets already in the buffer and goes to sleep.

The value of t_x should be set such that once a node into this state, if there is packet transmission is going on, the sensor has to be awake till the transmission is over and then still should be awake for some more duration to see if any node is sending an RTS to it.

7.2.4 Receive Transmit

A sensor in this state performs the tasks of receiving and transmitting packets. It should be observed that a sensor will be awake until it could forward the packet, after which it goes back to Ready to Receive state.

7.3 Performance of RAW

The performance of the protocol is verified on a standard network simulator ns2. Various scenarios were tested with MAC 802.11 at the MAC Layer with a simple Propagation Model. The simulations were carried on a 5RX5R network with a density of 6 nodes per RXR region and hence 150 nodes. The transmission radius can be varied accordingly for different network topologies. The model parameters and limits on transmission bit rates and energy ratings are set according to Crossbow MICA2 sensor nodes [20]. Nodes were deployed randomly in the rectangular region. The energy consumption for switching the radio from idle to sleep modes and vice versa is assumed to be negligible and hence not considered. The raw available bandwidth for each node is set to 1Mbps. The functionality of 802.11 is changed accordingly so that the node will be able to withstand sleep and idle schedules. The Short Retry Limit of MAC is changed to 2 and the packets drops for RTS failures at the MAC layer are eliminated. And are sent to the network layer with a retry limit of 5.

The simulations shown in Figure 7.2 are for a network of 150 nodes with a schedule period T_i being 0.5sec. There were 10 source nodes that generate packets at a date rate of 1-5 packets/sec and the performance is tested when there are one and two

destination nodes. It is observed that for a fixed density and with an increase in traffic, the awake time of the nodes does not vary much with the average latency being increased considerably. This can be seen in Figure 7.3.

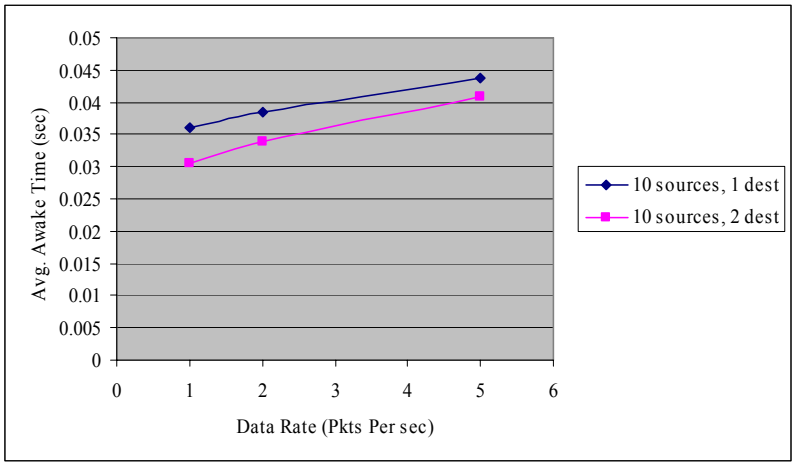

Figure 7.2 Effect of Awake Time of Nodes with Traffic

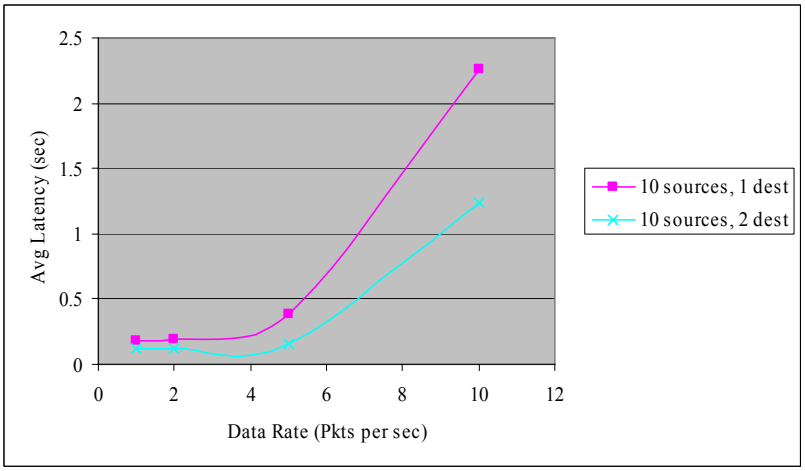

Figure 7.3 Effect on Latency with Traffic

The performance of the protocol is also seen when the schedule period changes. As seen in Figure 7.4, for a network size of 5RX5R with 500 nodes, density being 20 nodes per RXR region, the algorithm is tested for a schedule period of 0.25sec, 0.5 sec and 1.0 sec. Data is generated by one source at a rate of one packet/sec and is simulated

for 50 simulation seconds. The average latency is calculated for these simulations. All the parameters remain the same as noted above.

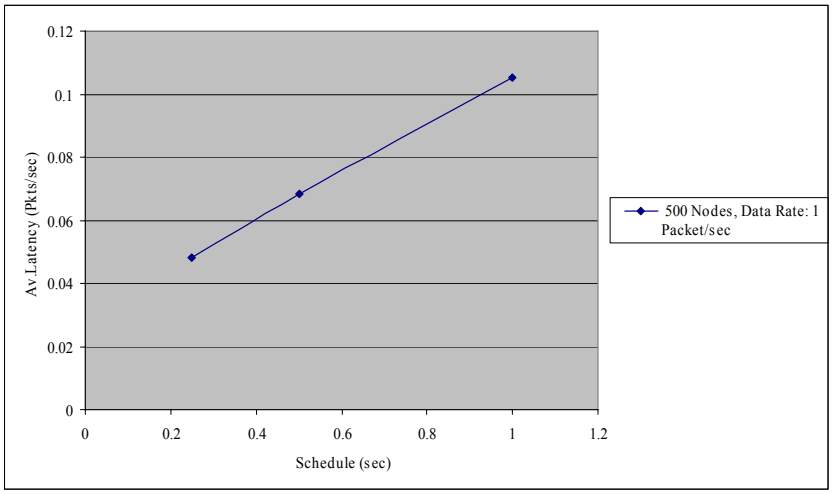

Figure 7.4 Effect of Schedule Period on the performance of protocol

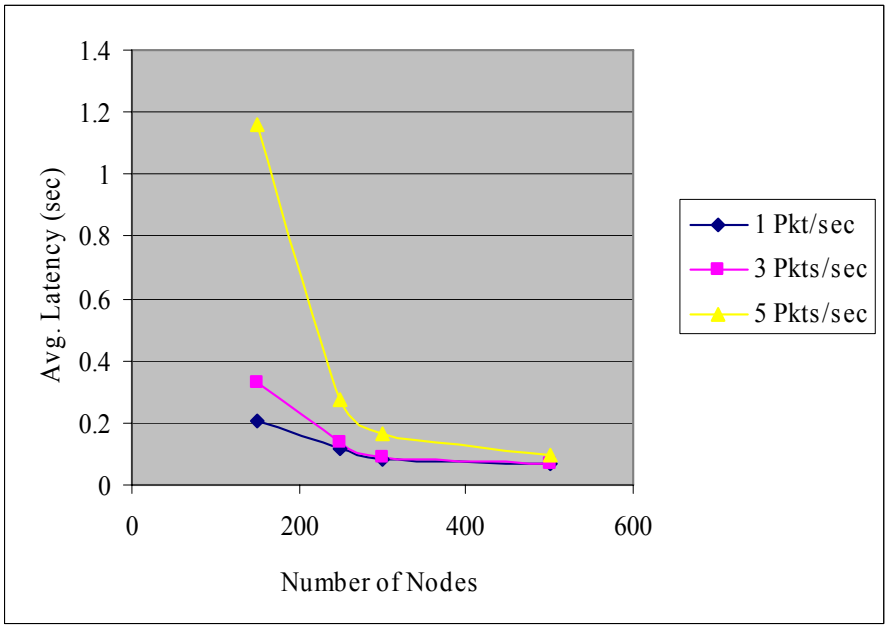

Figure 7.5 Performance of the protocol for various densities.

The simulations observed in Figure 7.5 shows that for a network of same rectangular region (5RX5R), when the density is increased from 4 nodes per RXR region to 10 nodes, 12 nodes and 20 nodes per RXR region, the average latency for the overall simulation is decreased and this shows the performance of the protocol for a network of very large densities. The scenario is tested for varying data rates from one packet/sec to five packets/sec. The schedule period is maintained as 0.5 sec for this set of simulations.

The average latency observed is for a packet delivery ranging from 98% to 50%.

CHAPTER 8: CONCLUSION AND FUTURE WORK

This thesis provides the implementation of various routing protocols for the Simulator developed at LSU. The comparisons of Directed Diffusion protocol in SensorSimulator with ns2 validates the implementation details of various modules developed in the simulator. The study of the various routing protocols with 802.11 adds to the modules developed for the simulator and enables the further analysis. This thesis also provides enxtensions to the 802.11 MAC Layer, Physical Layer and the Energy Module that are developed in the initial version. The critical task of the MAC layer to consider the sleep and idle switching of a sensor node is carefully designed as this being the necessary task for a sensor node in a sensor network application. Script files for collecting the routing protocol statistics are included in the simulator. The simulator and the support provided makes it very easy to develop and test protocols very fast and obtain results for large simulations at a reasonable amount of time. Simulations were carried for a sensor network of 2000 nodes and also with a density of 11 nodes per transmission region. This shows the scalability achieved by the simulator. The simulations show that the performance of simulator in terms of execution time remains the same for large number of nodes also. The comparisons made with ns2 validate this.

The simulation analysis shows that the algorithms Broadcast Protocol for Sensor Networks, Efficient Coordination Protocol for Sensor Networks and Random Asynchronous Wakeup Protocol are efficient in terms of energy and network life time. These implementations in the simulator expand the set of protocols developed for it.

REFERENCES

[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless Sensor Networks: A Survey. Computer Networks, 38:393–422, 2002.

[2] Chen, G., J. Branch, M. J. Pflug, L. Zhu and B. Szymanski (2004). SENSE: A Sensor Network Simulator. Advances in Pervasive Computing and Networking. B. Szymanksi and B. Yener, Springer: 249-267.

[3] Kevin Fall, Kannan Varadhan, Editors, The VINT Project, UC Berkeley, LBL, USC/ISI, and Xerox PARC, The ns Manual.

[4] OPNET Technolgies, Inc. OPNET Modeler.

[5] Ahmed Sobeih, Wei-Peng Chen, Jennifer C. Hou, Lu-Chuan Kung, Ning Li, Hyuk Lim, Hung-Ying Tyan, and Honghai Zhang, J-Sim: A Simulation and Emulation Environment for Wireless Sensor Networks.

[6] Dr.S.S.Iyengar, Vatsalya Kunchakarra, Ankur Suri, LSU-SensorSimulator, User Manual, Version 1, February 2005.

[7] A. Vargas. OMNET++ Discrete Event Simulation System, version 2.3 edition, 2003.

[8] C. Mallanda, A. Suri, V. Kunchakarra, S.S. Iyengar, R. Kannan, A. Durresi, Simulating Wireless Sensor Networks with OMNeT++ , submitted to IEEE Transactions on Computers.

[9] A. Tannenbaum. Computer Networks. Prentice Hall, 2002.

[10] A. Durresi, V. Paruchuri, R. Kannan, S.S. Iyengar, "Optimized Broadcast Protocol for Sensor Networks, IEEE Transactions on Computers, Volume 54, Issue 8, August 2005, pp. 1013 - 1024.

[11] G. Dommety and R. Jain. Potential networking applications of global positioning systems (GPS). Tech. Rep. TR-24, CS Dept., The Ohio State University, April 1996

[12] David Cavin, Yoav Sasson, Andre Schiper, Distributed Systems Laboratory, On the accuracy of MANET simulators

[13] David M. Nicol, Dartmouth College, Comparison of Network Simulators Revisited, May 20, 2002.

[14] “Geographical and Energy Aware Routing: A Recursive Data Dissemination Protocol for Wireless Sensor Networks” Yan Yu, Ramesh Govindan and Deborah Estrin UCLA Computer Science Department Technical Report UCLA/CSD-TR-01-0023, May 2001.

[15] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Transactions on Networking*, 1(1): 2–16, February 2003.

[16] Kershner, R., "The Number of Circles Covering a Set, "*Amer. J. Math,* 61, 665-671, 1939

[17] Jerry Zhao and Ramesh Govindan, "Understanding packet delivery performance in dense wireless sensor networks," Proceedings of the 1st international conference on Embedded networked sensor systems, Sensys 2003, Los Angeles, California, USA November 05 - 07, 2003, pp: 1 – 13.

[18] Alec Woo, Terence Tong and David Culler, "Taming the underlying challenges of reliable multihop routing in sensor networks, " Proceedings of the 1st international conference on Embedded networked sensor systems, Sensys 2003, Los Angeles, California, USA, November 05 - 07, 2003, pp: 14 – 27

[18] Vamsi Paruchuri, Shivakumar Basavaraju, Arjan Durresi, Rajgopal Kannan and S.S. Iyengar, Louisiana State University, Random Asynchronous Wakeup Protocol for Sensor Networks

[19] T. He, J. A. Stankovic, C. Lu, and T. F. Abdelzaher. SPEED: A stateless protocol for real-time communication in sensor networks. International Conference on Distributed Computing Systems (ICDCS 2003), May 2003.

[20] Crossbow MPR/MIB mote hardware users manual. www.xbow.com/Support/manuals.htm.

[21] Vamsi Paruchuri, Arjan Durresi, Vatsalya Kunchakarra, ECP: An Efficient Coordination Protocol for Wireless Sensor Networks.

VITA

The author, Vatsalya Kunchakarra was born in Khammam, Andhra Pradesh, India. She graduated from Excellent Junior College, Khammam, in 1999. In May 1999, the author attended Osmania University, Hyderabad, India, with a major in electronics and communications engineering and obtained Bachelor of Engineering degree in 2003. She continued her graduate study in systems science at Louisiana State University. As a member of sensor networking research laboratory she was actively involved in the design and development of sensor simulator and is currently a candidate for the degree of Master of Science in Systems Science.